A MODULAR LORAWAN INSPIRED INTERNET OF THINGS APPROACH TO COLLECTING SENSOR DATA VIA SOFTWARE DEFINED RADIO

AUTHOR:
Tristan Van Cise, B.S.

APPROVED:
Dr. Jon Genetti, Committee Chair
Dr. Orion Lawlor, Committee Member
Dr. Jonathan Metzgar, Committee Member
Dr. Chris Hartman, Department Chair
Department of Computer Science

A Project Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

University of Alaska Fairbanks

May 2020

Abstract

The emergence of simple Internet of Things (IoT) devices has habituated the ability to efficiently collect data and communicate information between devices with ease. Similarly, Software Defined Radio (SDR) has compacted radio communication into a USB dongle capable of receiving radio signals from most radio transmitters. In this approach, the ease of IoT device communication and versatility of SDR data collection and transmission techniques is combined to monitor building thermal decay. The system developed to collect thermal decay data is adapted from the Long Range Wide Area Network (LoRaWAN) IoT architecture and is designed to facilitate variable size collection environments and real-time data visualization. This paper will outline the implementation and capabilities of the collection system and highlight alternate applications and hardware implementations of the underlying framework.

Acknowledgements

I acknowledge the support from the US Army Program633734T1500, Military Engineering Technology Demonstration. I thank Dr. Tom Douglas, Dragos Vas, Heike Merkel, Matt Perry, Arsh Chauhan, Dayne Broderson, Bjorn Oberg, Angela Urban, Dr. Alexander Zhivov, and Dr. Jeremy Kasper for their assistance with the paper approval process and willingness to aid in development and deployment efforts during testing. I also thank Dr. Jon Genetti, Dr. Orion Lawlor, and Dr. Jonathan Metzgar for being on my committee and providing guidance throughout the development of this project and paper.

Contents

List of Figures

List of Tables

List of Abbreviations

IoT	**I**nternet **o**f **T**hings
SDR	**S**oftware **D**efined **R**adio
MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport
LoRa	**L**ong **R**ange
LoRaWAN	**L**ong **R**ange **W**ide **A**rea **N**etwork
LPWAN	**L**ow **P**ower **W**ide **A**rea **N**etwork
NB-IoT	**N**arrow**B**and **I**nternet **o**f **T**hings
SMA	**S**ub**M**iniature version **A**
RTC	**R**eal **T**ime **C**lock
UUID	**U**niversally **U**nique **I**dentifier
InfluxDB	**I**nflux **D**ata**B**ase
TIG	**T**elegraf **I**nfluxDB **G**rafana
NTPD	**N**etwork **T**ime **P**rotocol **D**aemon
DOM	**D**ocument **O**bject **M**odel
TCP	**T**ransmission **C**ontrol **P**rotocol
SSL	**S**ecure **S**ockets **L**ayer
AES	**A**dvanced **E**ncryption **S**tandard

Chapter 1

INTRODUCTION

With the projected increase in IoT devices to reach billions worldwide in the coming years, it is important to look at solutions that are scalable, easy to deploy, and affordable in order to enable wireless data collection efforts in numerous fields like microbiology, fisheries, and smart city development [12]. The collection system in this paper seeks to satisfy those requirements in addition to being modular enough to support a variety of sensors that perform an array of different functions. However, even though the system is capable of supporting different sensors, this paper will focus on a specific use case regarding the collection of temperature and humidity data from a building cooling down after the heat is turned off i.e. the thermal decay of the building [1] [3]. This simple and specific use case will illustrate the data collection capabilities of the system and provide a foundation for pairing system capabilities with real world applications. Later, the system's application with different sensors and data transmission techniques to enable outdoor usage via Long Range (LoRa) Low Power Wide Area Networks (LPWANs) is discussed.

1.1 LoRaWAN Architecture

The system's architecture is adapted from the LoRaWAN architecture which is split into four primary sections: end nodes, concentrators, network server, and application server. The end nodes are individual sensors that perform actual measurements like GPS or temperature and transmit data via chirps on 433MHz. End nodes make up the granular portion of most applications, so there are usually several clusters of end nodes applied to an environment at a time. In each of these clusters, a concentrator is positioned to receive chirps from the end nodes before sending the data to a central network server via cellular or ethernet connection [7]. Data is then stored in a central database accessible by a network to allow applications and other servers easy access for visualization and analytic.

1.2 Software Defined Radio (SDR)

SDR technology is a central component that bridges the communication gap between end node data and concentrators receiving data. An SDR USB dongle resides on each concentrator and is responsible for intercepting end node chirps and forwarding the data to *rtl_433*, a decoder library by Benjamin Larson [8] for devices that chirp on 433MHz. This library performs all sensor data decoding and

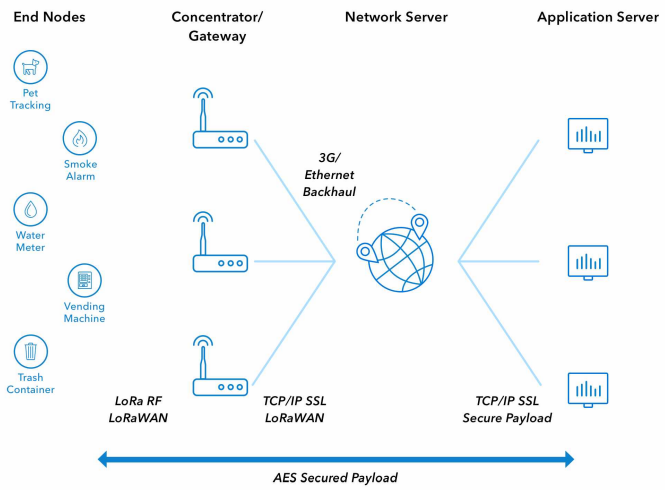

FIGURE 1.1: LoRaWAN architecture used to funnel data from individual sensors to a central device [5].

is the reason why the system as a whole can accept data from a number of different sensors.

1.3 Message Queuing Telemetry Transport (MQTT)

MQTT is a publish subscribe message passing framework used to handle data transfers between devices in the architecture. The publish subscribe design pattern used by MQTT is centered on the interaction between publishers and a broker. The concentrators primarily play the publisher role in the system because after decoding data from an end node they publish data to a network server, which acts as the primary broker. In order for a publisher to publish data on the broker, it must choose a topic that the broker is hosting to send data to. For example, if a concentrator wanted to publish data from an end node responsible for GPS coordinates to the broker, it may publish the data to the /GPS/end_node_id topic. Topics are a method for organizing data flow to the broker so that subscribers, applications that use the data received by the broker, can retrieve new data when it arrives in an organized fashion. In terms of the system, the applications and visualization components of the architecture are subscribers that can subscribe to specific topics like /GPS/# to receive all GPS data (# is an MQTT wildcard character). Notably, MQTT does not provide any data storage, it only provides a medium by which data can be asynchronously sent and received.

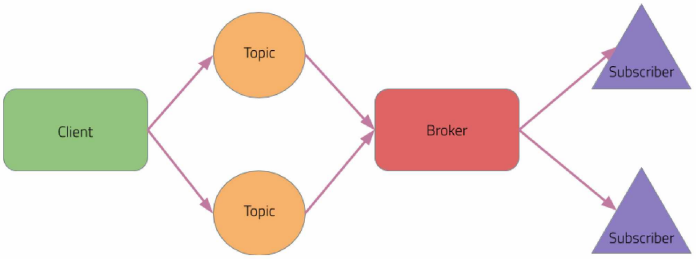

FIGURE 1.2: The MQTT data pipeline for sending data from a client to a broker for subscribers to use.

Chapter 2

RELATED WORK

Researchers have been working on finding various compact solutions for facilitating the increased need for IoT devices that handle complex problems in what some call the fourth industrial revolution [12]. As with all large platforms, a stable infrastructure is required to enable devices to communicate with each other securely. In smart cities and companies that utilize industrial IoT devices, software defined solutions such as software defined radio and software defined networking offer programmatic flexibility that automates and replaces many functions previously handled by hardware [4].

IoT distributed systems combined with LoRa have proven to enhance traditional measurement systems performing similar functionality, particularly in aspects like range and wireless monitoring [9]. Some applications of such systems are renewable energy monitoring [2], public safety communication, and resource monitoring [11]. LoRa is one of many LPWAN technologies that researchers use to enable IoT development because of its ability to provide a range of different communication methods to satisfy high bandwidth or long range constraints. Most of the time, sensory IoT devices are applied to areas where distance is more of a concern than data rates [6], which is what LPWAN technologies like LoRa and NB-IoT focus on providing [10]. However, the effectiveness of an LPWAN technology hinges greatly on the environment in which it is deployed in. Studies have shown that although cities are moving towards smarter IoT sensors, LPWAN technologies behave less effectively in dense urban areas due to signal obstruction from buildings and other man made structures [13].

Chapter 3

METHODS

The system predominantly follows the LoRaWAN architecture by implementing the end node and concentrator sections, however it differs by combining the network server and application server sections together. This decision was made in order to compact the system and reduce the amount of communication channels required to perform data analytic and visualization. Effectively, this reduces the four stage LoRaWAN architecture to a three stage architecture consisting of end nodes, concentrators, and the new combined stage called the centralizer. The following sections describe the functionality of each of these three stages before giving an in depth look at the individual components that make up each stage and how they pertain to thermal decay data collection.

3.1 Stage 1: End Nodes

An assortment of temperature, humidity, and consumer grade weather stations were used to make up the end node stage. These individual sensors are the foundation of the thermal decay data collection and due to their simple nature of chirping data on a set time interval, deploying them for a test is very plug and play. The Acurite Tower Sensor and Acurite 5n1 Weather Station were the two primary sensors used for tests. The tower sensors only measured temperature and humidity while the 5n1 Weather Station measured temperature, humidity, precipitation, wind speed, and a few other measurements. Each sensor chirped on an approximate fifteen second interval and were configured to use a combination of a unique IDs and channels to disambiguate measurements between sensors.

3.2 Stage 2: Concentrators

Concentrators are centrally located devices that are positioned in clusters of end nodes. Usually a concentrator is responsible for 20-30 end nodes and is capable of receiving data from sensors in a 100-200 foot radius depending on location, material interference, and hardware components on the concentrator itself.

A concentrator is built from a Raspberry Pi 4 with 4GB of RAM and utilizes a SDR and Real Time Clock (RTC) to receive data and accurately produce timestamps. The SDR unit itself is simply a USB dongle with a SMA female connector for screwing on various types of monopole or dipole antennas.

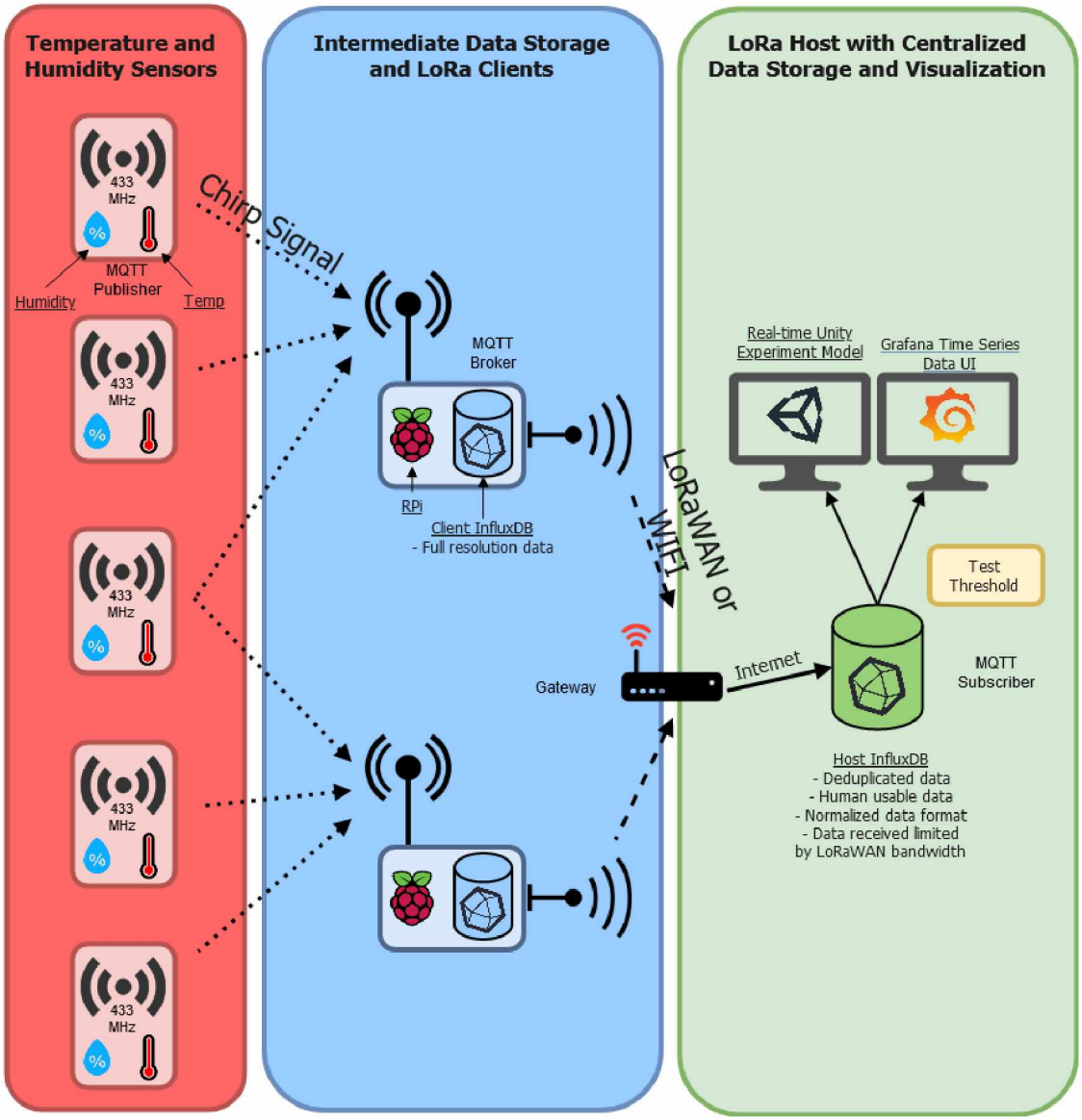

FIGURE 3.1: The data pipeline and system architecture of the three stages. The red section corresponds to the end nodes, the blue section corresponds to the concentrators, and the green section corresponds to the centralizer. The data pipeline from the end nodes to the centralizer is shown using dashed arrows.

Concentrators are network connected with static IPs via a mesh network that is designed to be extendable and span across an entire test area. Due to the sheer volume of sensor readings an individual concentrator may receive, the mesh network is used solely to establish a reliable communication channel between a concentrator and the centralizer. Alternative options such as sending data over radio from the concentrator's SDR would consolidate the amount of deployment hardware needed to use the system, but a network makes communication between devices much simpler and interfaces better with libraries like Mosquitto which has a web API for performing MQTT operations. A network also allows for easy expansion in the event a single router is not enough to cover an entire test zone, in which case a mesh network is used. For our purposes, the primary router used to establish the network is a Linksys E2500 router which if applicable, is extended

(a) Acurite tower sensor. See appendix A.2 for device accuracy and precision.

(b) Acurite 5n1 weather station. See appendix A.1 for device accuracy and precision.

FIGURE 3.2: End nodes used for obtaining thermal decay test data.

FIGURE 3.3: A fully assembled concentrator Raspberry Pi 4.

(a) RTL-SDR

(b) Nooelec-SDR

FIGURE 3.4: The two types of SDR dongles used on the concentrators.

(a) Linksys E2500 Router (b) TP-Link Travel Router

FIGURE 3.5: Mesh network hardware.

into a mesh network using TP Link Travel Routers configured in range extender mode.

3.2.1 Concentrator Architecture

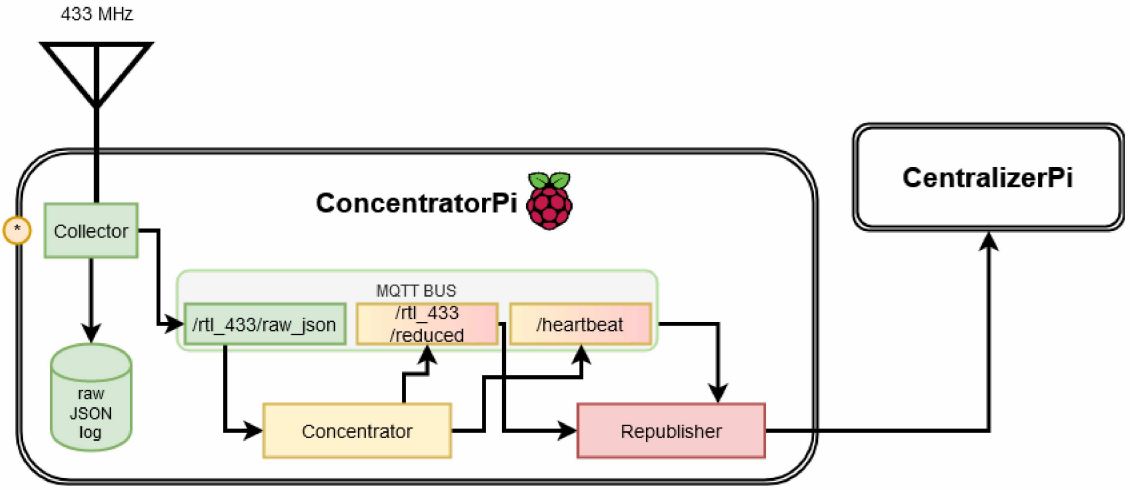

FIGURE 3.6: Concentrator architecture and data pipeline.

The architecture of the concentrator is composed of three services (collector, concentrator, and republisher) and an MQTT bus for data management. Systemd starts each of the services when booting and in the event of a network, power, or hardware outage, each service is configured to restart and reconnect when the problem has been resolved.

At the beginning of the concentrator's data pipeline the collector service receives end node chirps on 433MHz via a SDR and decodes the packet into a JSON formatted dictionary using the rtl_433 library. At this point, a timestamp is injected into the JSON using the time provided by the RTC in order to capture the most accurate time a chirp was received. This is done in the collector service to prevent timestamp lag from data processing performed by other services later in the data pipeline. The collector service then writes the raw end node data to a local file stored on the concentrator and publishes the data to the /rtl_433/raw_json topic on the MQTT bus.

For thermal decay data, a small timestamp inaccuracy is insignificant when looking at the data as a whole, however, for more time sensitive applications with faster data intervals, a timestamp lag could lead to invalid data. Therefore, having the collector service inject the timestamp is the solution to keeping the system modular and applicable to a wide range of tests.

Part of the transmission protocol of some end nodes (primarily the Acurite Tower Sensors) is to send a burst of chirps every 15 seconds. This results in receiving duplicate data and performing more write and publish operations to the log file and MQTT bus per end node chirp. These operations can be very taxing for a Raspberry Pi and effectively multiplies the amount of data the concentrator needs to process per end node by the number of times it chirps. For this reason, the collector service additionally monitors the incoming data from the SDR and if it sees a duplicate within a short period of time it discards it.

After the collector service publishes the raw end node data to the MQTT bus, the concentrator service, which is subscribed to /rtl_433/raw_json, begins formatting it. An example of a typical JSON string from the collector service is shown in Figure 3.7.


```
{
    "time": "2020-02-27 01:02:21",
    "model": "Acurite-Tower",
    "id": 14654,
    "channel": "A",
    "battery_ok": 1,
    "temperature_F": 68.000,
    "humidity": 1,
    "mic": "CHECKSUM"
}
```


FIGURE 3.7: An example of the JSON end node data received from the collector service after it is decoded.

Using a combination of the model, id, and channel fields in Figure 3.7, the concentrator service generates a Universally Unique Identifier (UUID) for each end node. Note that this UUID generation only works for end nodes that contain those fields in the data they chirp. For our example of thermal decay, we determined the three aforementioned fields would be sufficient to generate a UUID for each sensor.

With the end node UUID constructed, the concentrator service combines it with the original end node data and a concentrator unique identifier to create a data packet that can be associated with a specific end node and concentrator. This new data packet is then published to the /rtl_433/reduced topic on the MQTT bus. The concentrator service also has the option of rate limiting the amount of new data that can be posted to the /rtl_433/reduced topic per end node. This works by inputting a time interval and keeping track of the last time an end node chirped data. If a new chirp from an end node is received and the time difference from the last time it chirped is less than the time interval the data is discarded, otherwise, the data is published to /rtl_433/reduced.

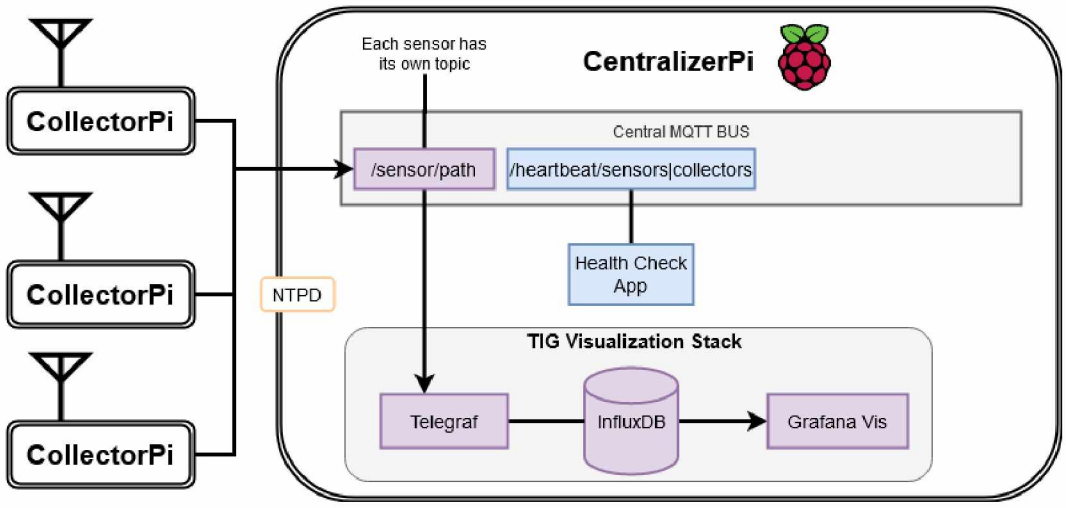

FIGURE 3.8: Centralizer architecture and data pipeline

The concentrator service also publishes a data packet containing the end node UUID and concentrator unique identifier to a topic called /heartbeat, which is used later to track end node and concentrator up-time.

The /rtl_433/reduced and /heartbeat topics are subscribed to by the republisher service that simply acts as a data forwarder from the concentrator to the centralizer once all preliminary data processing is complete. The republisher is the only service that communicates directly with the centralizer and as such, is responsible for knowing where the centralizer exists on the network and maintaining a reliable connection with it.

3.3 Stage 3: Centralizer

The centralizer is the final data destination in the pipeline and acts as the central collection and visualization component of the system. Multiple concentrators existing on the same network send end node data to the centralizer, which is running a time series database. The centralizer is broken into two main parts which each have subcategories. The first part of the centralizer is the architecture and data pipeline that routes end node data to the second part of the centralizer which handles visualization.

3.3.1 Centralizer Architecture

When the concentrator's republisher service publishes data to the centralizer, it uses a unique topic for each end node UUID. For example, the topic path /end_node/14654/A would correspond to an end node with a sensor ID of 14654 sending data through channel A. By combining the sensor ID and the channel it chirps on, the UUID topic /end_node/14654/A is formed. This makes data retrieval easy for visualization applications because it allows them to query a specific sensor or group of sensors depending on how the topic system is setup. For example, if topics were split up based on floor, then sensor id, then channel, a subscriber could query the /floor1/# topic to get all sensors on floor one.

The only other topic that resides on the centralizer MQTT bus is /heartbeat, which is subscribed to by the health check app. As previously mentioned, the centralizer monitors the up-time of each end node and concentrator through /heartbeat. This is done by caching the timestamp of the most recent end node chirp and configuring a timeout constant. If an end node fails to chirp new data to the concentrator that previously received it within the number of seconds specified by the timeout value, it is marked as offline.

To make viewing the system health easily accessible, the centralizer also runs a lightweight web app that shows the online status of each concentrator and the end nodes it is receiving data from. The web app is a single page application built using a progressive web framework called Vue. Vue uses a data binding system called v-models to seamlessly map changes in data to frontend DOM rendering. When the concentrator republisher service publishes to /heartbeat on the centralizer, the centralizer receives that data and writes the end node status to a file. The contents of that file are bound using v-models meaning if the file state changes, the DOM automatically re-renders the page without requiring a page reload.

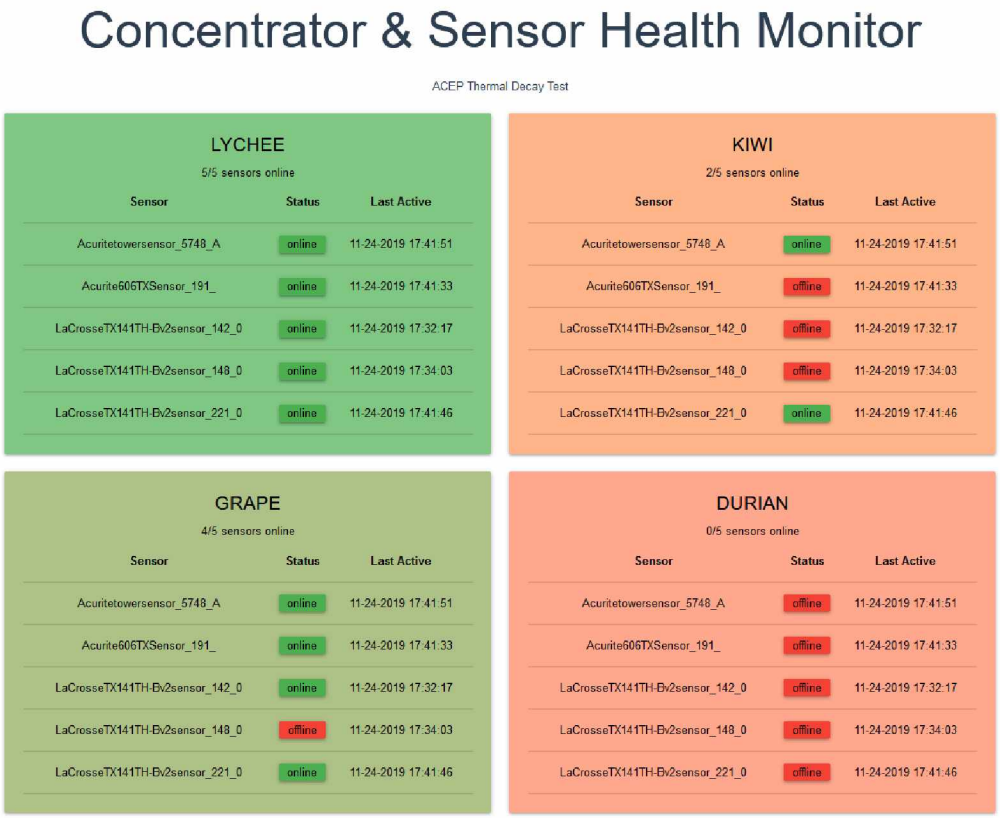

FIGURE 3.9: The web based Graphical User Interface (GUI) for viewing real-time end node and concentrator statuses. End nodes toggle between a green online state and red offline state based on the amount of time that has passed since a new reading has been received. The concentrator statuses are dynamically colored based on the number of online end nodes. More online end nodes makes the concentrator health color appear greener and more offline nodes makes the concentrator health color appear redder.

3.3.2 Centralizer Data Visualization and Storage

The most important function of the centralizer is storing all end node data in a time series database so it can be accessed later and used by visualization applications. The system that manages the transfer of data from the end node UUID topics on MQTT to the database and visualization applications is called the TIG stack. TIG stands for Telegraf, InfluxDB, and Grafana, which are the three services that retrieve end node data from MQTT, store it in a database, and produce real-time visualization GUIs. Each service runs in a Docker container due to the immense amount of configuration that has to be done in order for each service to behave properly. Containerization also grants an added layer of security which is particularly important for the TIG stack because it contains the central database with all test information.

The first stage of the TIG stack is Telegraf. Telegraf directly interfaces with all the end node UUID topics on MQTT and prepares batch insertion queries to send to InfluxDB. Database insertion operations are performed in batches to limit the amount of asynchronous overhead time of sending sequential insert commands. Telegraf also manages the tables and naming scheme in InfluxDB by automatically creating tables that correspond to each end node UUID. Organizing the database in this fashion makes it more accessible to applications and normalizes the database querying scheme.

Because InfluxDB is a time-series database, timestamps are used as primary keys for each row when insertions are performed on individual tables. The fields in the tables are simply the same fields from the end node JSON, therefore as long as valid UUIDs are being generated, each table will contain data from the same end node with identical fields for the duration of an entire test.

In reference to the duplication problem handled by the concentrator's collector service, the centralizer is faced with a similar issue but instead of receiving duplicate end node data from the end nodes themselves, it receives duplicate data from concentrators. This duplication problem occurs due to multiple concentrators being in range of receiving chirps from the same end node. The concentrators then publish the duplicate data to the centralizer, which batch queues two database insertions with the same primary key.

InfluxDB handles these primary key collisions by simply replacing data with the same primary key and for our purposes, this sufficed as a method for dealing with duplicates. Due to the long intervals between end node chirps, batch insertions, and the small amount of overlap between concentrators during tests, the InfluxDB replace operation did not add any notable overhead to the centralizer. However, this solution is not sufficient for handling tests where concentrators have large areas of overlap and are collecting data from end nodes at a much higher rate.

The final stage of the TIG stack is the Grafana web GUI. Grafana is a framework that inputs time-series data and create dashboards that graphically convey the behavior of the data over time. There are a number of different graphical tools that Grafana uses to create dashboards but for our purposes, we simply created a default dashboard that displays all thermal decay readings per end node on a graph. Grafana is extendable and is capable of adapting its interface for all sorts of different data visualization requirement. This is especially valuable for the modularity of the system because it grants end users the ability to create their own data visualization mediums for an array of different environments. For example, if

a test takes place in a three story building, an end user may want to create graphs on a dashboard that correspond to sensor reading on each floor. For our tests, we used this functionality to monitor critical rooms in buildings such as boiler rooms and server room, which have sensitive temperature and humidity requirements that if not satisfied, could damage the equipment contained within.

Grafana also has the ability to generate dashboards given a configuration file. Programmatically, this is advantageous to mapping database queries to end nodes because a configuration file can be generated while the end nodes are being deployed and simply loaded into Grafana to provide a base dashboard for end users to work with. It being web based, Grafana can also be configured to be accessed outside of a local network or hosted on a password protected website for end users to view at any place any time. Additionally, Grafana features an alerting system that can trigger scripted events to alert users that something is wrong, a condition has been satisfied, or send commands to fix a malfunctioning device.

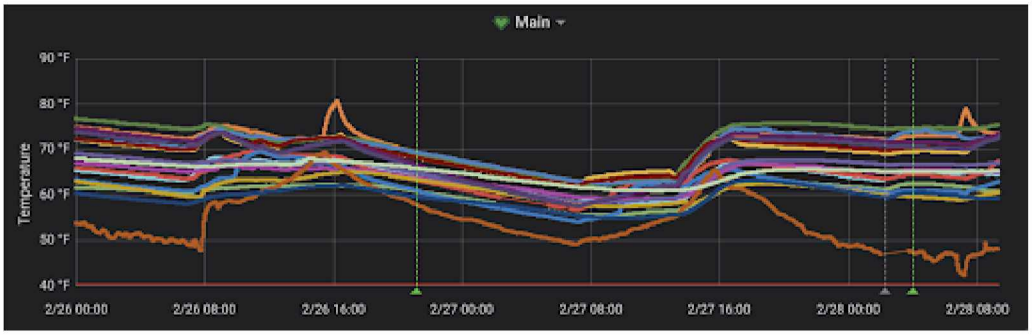

FIGURE 3.10: A Grafana dashboard displaying end node temperature data on a time-series graph.

3.4 Deployment

A typical deployment for a thermal decay test consisted of the following procedure:

1. Set up mesh network nodes and verify connection is obtainable from all areas of the building.
2. Power on the centralizer, connect it to the mesh network, and verify its services are running.
3. Place end nodes all throughout the interior and some outside, ensuring all rooms and critical building areas are measurable.
4. Deploy concentrators in central locations throughout the building such that each end node can have its chirps received.
5. Verify that each end node and each concentrator is accounted for on the centralizer health check GUI and ensure data is correctly being sent through the pipeline.

For smaller buildings that do not need a mesh network and multiple concentrators to provide full coverage, the concentrator and centralizer can be combined into one Raspberry Pi. This reduces the number of steps in the data pipeline to

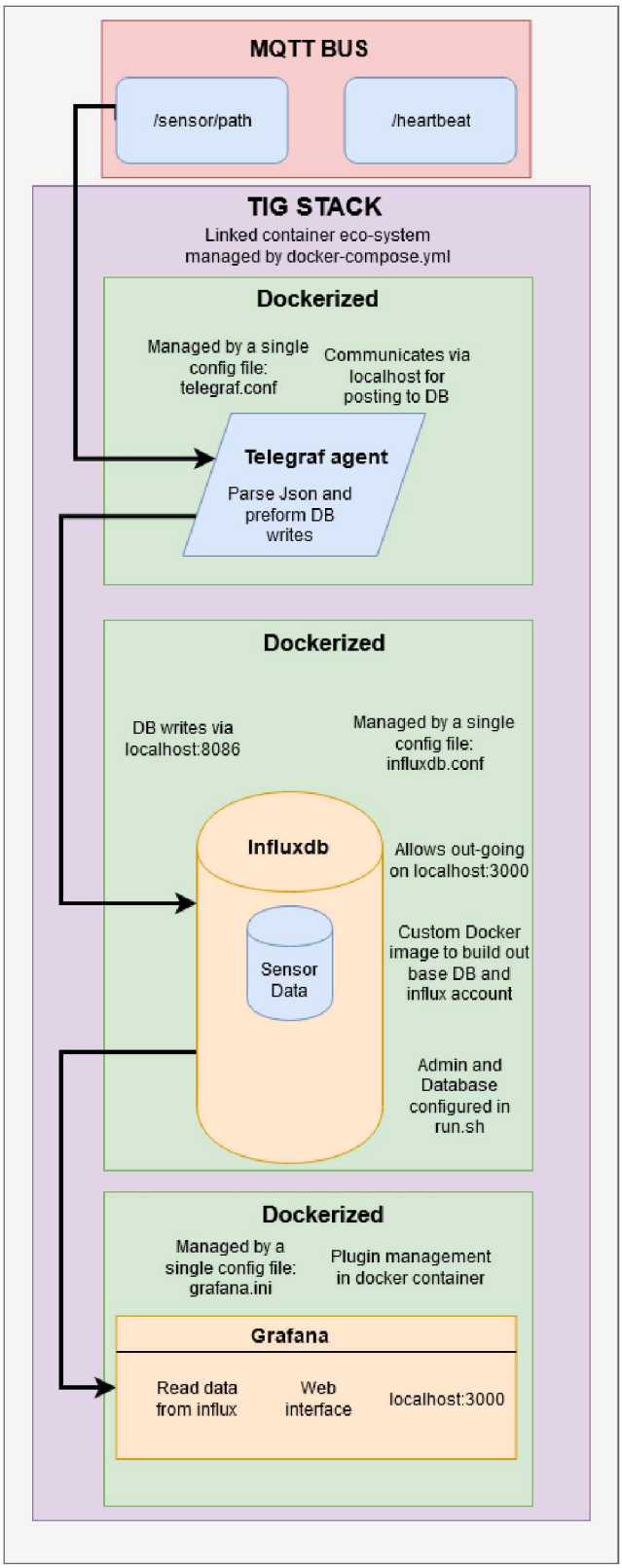

FIGURE 3.11: The Telegraf, InfluxDB, and Grafana stack for storing data from MQTT and producing real-time visualization via browser.

two because the end nodes chirp directly to the centralizer. In terms of architecture, the combined concentrator/centralizer shares one hybrid MQTT bus and combines the collector and concentrator services on the concentrator with the health check GUI and TIG stack on the centralizer.

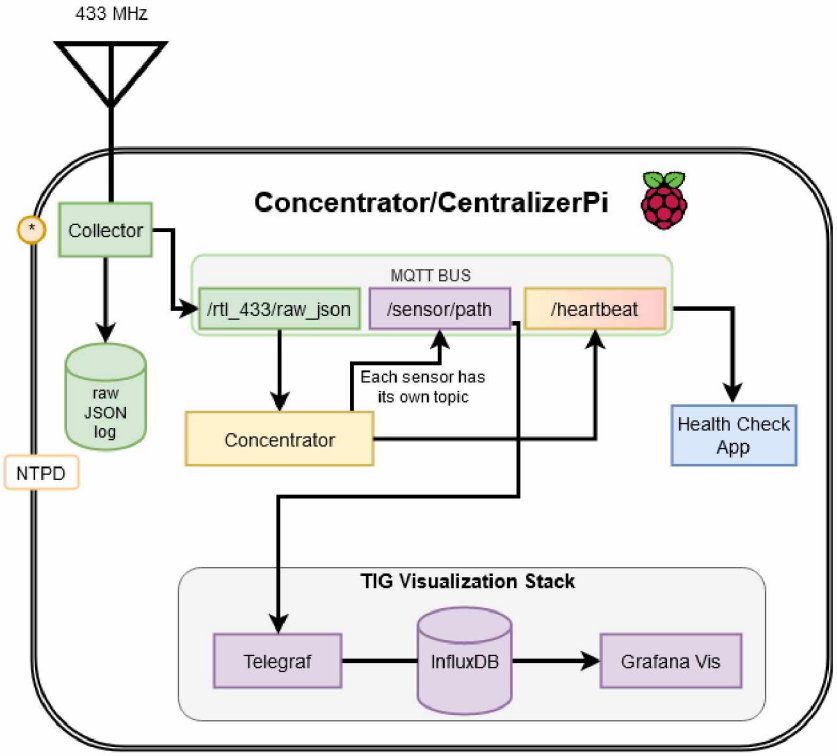

FIGURE 3.12: The architecture diagram of the combined concentrator/centralizer Raspberry Pi. Data is received like usual through the collector service but is published to the end node UUID topic after it is received by the concentrator service instead of /rtl_433/reduced; the /heartbeat topic remains unchanged. From the end node UUID and heartbeat topics, the TIG stack and health check GUI subscribers receive the data and handle it as they would on the centralizer.

3.5 Installation

After assembling the hardware, an installation script is used to configure any Raspberry Pi 3 or 4 to operate as a concentrator, centralizer, or both. Configuring a Raspberry Pi involves downloading Python dependencies, setting up library files, and deploying custom systemd services. Depending on the function of the Raspberry Pi, the installer also asks for network configuration information, credentials, and identifiers for some services. To setup a Raspberry Pi as both a concentrator and centralizer the script is simply ran twice, once for each system functionality. After installation, the Raspberry Pi is operational and ready to be transported or used in a test at any time.

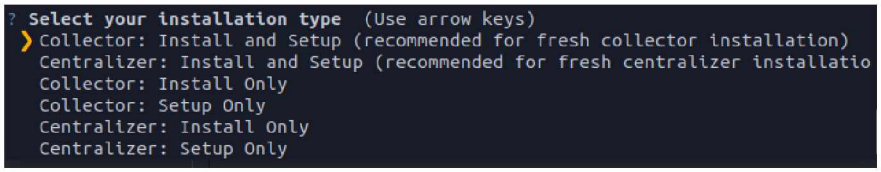

FIGURE 3.13: The initial menu of the install script used to setup a Raspberry Pi to be a concentrator, centralizer, or both. The "install only" and "setup only" options download the dependencies or prompt the user for configuration information depending on the installation type.

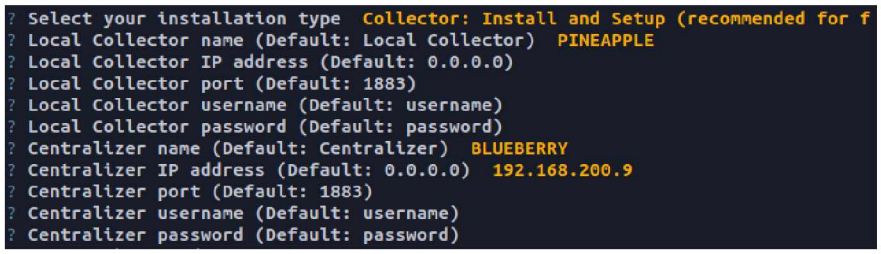

FIGURE 3.14: Example full installation prompt for a collector.

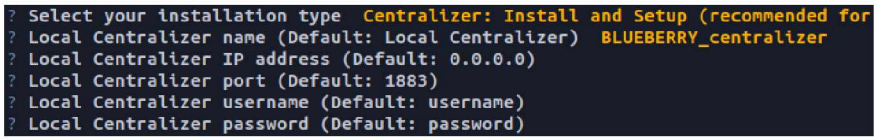

FIGURE 3.15: Example full installation prompt for a centralizer.

Chapter 4

RESULTS

A total of five tests in three different buildings were conducted to judge system performance and collect thermal decay data. Each test ran between 8-48 hours and the buildings in which the system was deployed differed in aspects like number of floors, insulation material, and number of rooms. Each factor of the building played a significant role in the overall system performance and dictated the number of hardware components needed to perform a successful full coverage test.

In the following sections each building will be referenced by number and a description of the architecture and internal layout will be provided alongside statistics on the number of end nodes and concentrators deployed. End nodes in this context refers to the temperature and humidity sensors which were deployed in important parts of the buildings. Typical placement locations included ceilings, next to vents, windowsills, interior and exterior walls, and the floor. Number of sensors and their placements were decided based on where the most temperature fluctuations would occur given the room layout.

4.1 Building 1

TABLE 4.1: Building 1 architectural and system deployment details.

Stories	3
Square Footage	11,000
Rooms	59
Concentrators	2
Centralizers	1
Mesh Network Nodes	1
End Nodes	50

Building 1 was a compact average height structure whose interior was primarily made of concrete and metal. The basement floor housed a garage and large boiler room full of metal pipes and concrete floors, walls, and ceilings. The second floor contained the primary entrance to the building and consisted mostly of small office sized rooms. The third floor was laid out in a similar fashion, however, it contained larger rooms and a small metal server room. A concrete stairwell was also located in the middle back of the building, which provided access to all three floors.

The E2500 router was situated on the second floor in the middle of the stairwell and was barely capable of transmitting a signal to the basement and third

floor hallways. Approximately 50 temperature and humidity sensor end nodes were deployed for the test, however most of the sensors resided in the basement, garage, and server room. For the office spaces and other rooms that contained sensors, an average of one sensor was used per room. One concentrator and the centralizer was stationed on the third floor in an open central living room. The other concentrator was positioned in the basement in an opening right outside the boiler room. Connections to both the concentrators was fairly spotty, averaging about two bars for the basement concentrator and one bar for the concentrator and centralizer located on the third floor.

The test ran for eight hours and successfully collected enough data to produce a cohesive data set. However, due to the spotty network connection to the concentrators, we noticed small blips in the time series data where large numbers of sensors would infrequently stop reporting for a couple intervals. The positioning of the concentrators captured all sensors deployed in the building but some sensors, particularly the ones in the boiler room and garage that were surrounded by concrete and metal, would occasionally go offline due to building materials blocking chirps. Miraculously, the concentrator located on the third floor was able to intermittently receive upwards of 44/50 sensors and most of the sensor it could not receive data from were in the metal enclosed boiler room. Had the building contained less metal and concrete, the system certainly could have maintained minimal coverage throughout the entire building using a single concentrator/centralizer Raspberry Pi.

4.2 Building 2

TABLE 4.2: Building 2 architectural and system deployment details.

Stories	2
Square Footage	23,000
Rooms	51
Concentrators	2
Centralizers	1
Mesh Network Nodes	3
End Nodes	80

Building 2 was a long two story building whose interior primarily consisted of wood with vinyl flooring and tiled ceiling panels. The first floor consisted of three large classrooms and a large office space with a few enclosed office rooms. The smaller second floor was occupied primarily by two sections of office cubicles and housed the boiler room, electrical room, and a server room, which were all made of metal and concrete. In the middle of the building, a narrow stairwell connected the two floors.

The E2500 was positioned at the base of the first floor stairwell and due to the lengthy floor area, was unable to broadcast a strong enough signal to be received in the test locations on the first and second floors. For this reason, two range extenders were introduced: one on the first floor to reach the office space and the second in a central kitchenette area between the two office spaces on the second floor. About 80 sensors were deployed for this test, however, due to the increased amount of open space in this building compared to Building 1, sensor placement

throughout the building was less dense and therefore required less sensors than would be expected despite the square footage increasing by a factor of two. The highest density of sensors resided on the second floor in and around the boiler, electrical, and server rooms. For the open office spaces, four sensors were deployed to each corner and two to three were scattered throughout the middle cubicles. Classrooms received two sensor due to their large size and exterior walls with windows, which cooled faster from outside winter temperatures.

Two concentrators were positioned in central locations on the first and second floor and were outfitted with long dipole antennas capable of collectively retrieving data from all the sensors. The centralizer was setup in a classroom on the first floor.

The tests in this building ran for 24 hours and again the system was able to collect sufficient data to produce a valuable data package. However, a few unexpected anomalies occurred while we were verifying whether all sensors were detected. The classroom walls were lined with a thin metal sheeting, which created a Faraday Cage that blocked sensors from transmitting data out of the room. To solve this, the sensors were repositioned in front of an open door to the classroom to allow for a transmission path that would avoid the Faraday Cage. On the second floor, a similar issue with the concrete and metal present in the boiler, electrical, and server rooms made retrieving data from those sensors particularly difficult. To remedy this problem, the concentrator was simply moved closer to those rooms and still maintained coverage of the entire second floor due to it being clearer and smaller than the first.

4.3 Building 3

TABLE 4.3: Building 3 architectural and system deployment details.

Stories	1
Square Footage	9000
Rooms	6
Concentrators	1
Centralizers	1
Mesh Network Nodes	1
End Nodes	17

Building 3 was the simplest of the three buildings because of its small amount of rooms and large areas of open space. The interior of the building was built with wood and the exterior was encased in metal panels. A majority of the interior was a high ceiling room filled with four foot filing cabinets in addition to a couple office spaces and small rooms. The boiler room was only accessible from the outside of the building and was separated from the inside by a concrete wall.

The small size of the building allowed us to use only the E2500 to create a network that would cover all testing areas. The 17 sensors were deployed in each room and spread across the filing cabinets, windowsills, and ceiling in the main open space room. A single sensor was placed in the boiler room and two were placed outside in addition to an Acurite 5n1 Weather Station for reading external temperature. A combined concentrator/centralizer unit was setup for this test and positioned next to the E2500 router in the center of the large room.

This test ran for 48 hours and even with the additional processing power required for the Raspberry Pi to run the concentrator and centralizer at the same time, most of the data was received and stored perfectly. The only inconsistency that was present in the test was the sensor located in the boiler room. Due to the material surrounding it, the sensor would often go offline for several chirps at a time, sometimes upwards of 10-15 minutes. Although, because the test lasted an extended period of time, enough data was collected to produce the graphs shown in Figures 4.1, and 4.2.

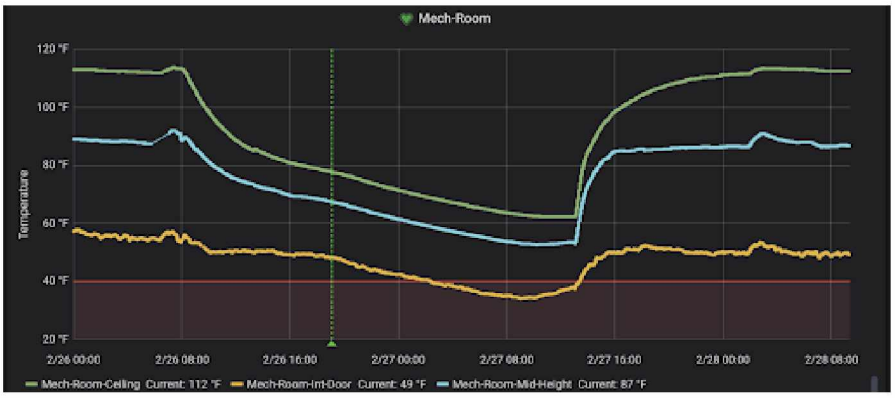

FIGURE 4.1: A customized graph showing the thermal decay of the boiler room

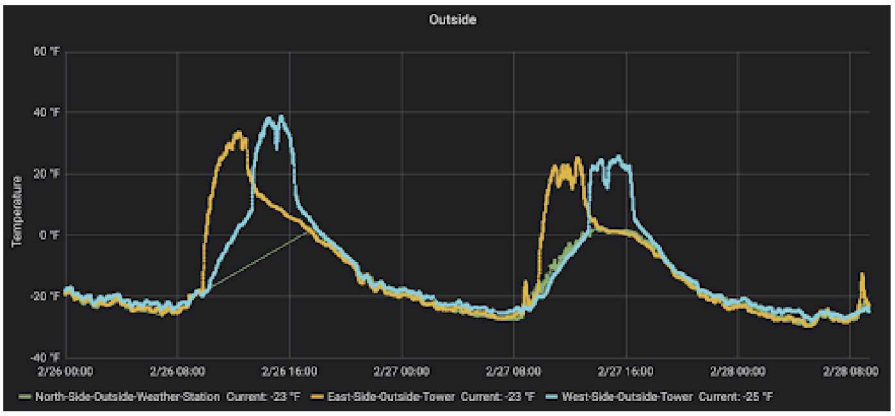

FIGURE 4.2: Another customized graph showing the outside temperature over 48 hours.

Chapter 5

ANALYSIS

The system in its current state is limited to indoor usage because a network is required to communicate information between a concentrator and the centralizer. The routers and range extenders are not built for outdoor use and operate best in an indoor scenario where wall power is regularly available. Thus, in addition to building weatherproof hardware cases, in order to adapt the system to outdoor use, a different method of communicating between concentrators and the centralizer must be developed.

Many communication avenues exist for sending data between devices without a local network such as satellite or cellular. However, both are relatively expensive and either require costly hardware or data plans to use. Particularly for cellular, not all remote areas of the world have reliable cell coverage, which can be a problem for field researchers who wish to collect data from said locations. Of the two options, cellular is the cheapest method to use, but in order to apply the system fully to outdoor usage, a method for artificially extending the range of cellular has to be implemented.

This is where the concept of a LoRaWAN comes in. Instead of using a local network or a cellular data plan to establish a communication channel between concentrators and the centralizer, a LoRaWAN could be used to either relay data directly to the centralizer or to a gateway that could upload the data to the centralizer. In terms of what this would look like on the system architecture diagram in Figure 3.1, data transfer between the blue concentrator stage and the green centralizer stage would be accomplished by a LoRaWAN or cellular and LoRaWAN combination.

In the former, the centralizer would act as a LoRaWAN gateway and the concentrators would upload data to it using LoRa. This application would suit the needs of an experiment where an environment has no cellular connection, but has access to power at some form of a base camp. The centralizer would reside at the base camp and receive data from remote concentrators in a five kilometer radius (more or less depending on the environment).

In the later case, the centralizer would be a cloud computer that receives data from cellular connected LoRaWAN gateways that communicate directly with concentrators in the field. From the perspective of a concentrator, the data pipeline for this method consists of sending end node data to a LoRaWAN gateway, which proceeds to forward the data to other gateways until a cellular connected gateway is reached. At this point, the data is uploaded to the cloud centralizer for application usage. An example use case of this method would involve a scenario

where cellular connection is available on site, but the area that the system needs to be applied to does not receive a cell signal (e.g. a cayou and Jeremy suggestedve or a ravine). This would require a LoRaWAN gateway to acts as an intermediary between concentrators and data being uploaded to the cloud centralizer via cellular.

Enabling this system to perform outdoor sensing would greatly increase the scope of the experiments and measurements it could be used for. Some of these measurements include:

- Soil moisture
- Plant growth
- Wildlife populations
- Flow rate in bodies of water
- Mineral contents
- Natural resource monitoring
- Animal tracking

See appendix B.1 for a list of more compatible sensors.

Chapter 6

CONCLUSIONS AND FUTURE WORK

The goal of this project was to produce a modular collection platform using software defined radio capable of being expanded and applied to a number of different fields of study. The system satisfies these requirements through each of its individual components that make up the architecture and data pipeline. Concentrators are capable of receiving any type of sensor data broadcast on 433 MHz via SDR and the system as a whole is expandable in its ability to scale up or down for any environment by adding, combining, and removing hardware components such as routers, concentrators, and end nodes. Ease of deployment is also achieved by eliminating the need to configure individual end nodes and limiting the amount of configuration required to setup concentrators and a centralizer. Similarly, data visualization is customizable to the needs of users and interfaces can be created on the fly to adapt the system to satisfy requirements. Redundant data storage on the concentrator and centralizer also guarantees that in the event of a system malfunction, there are failsafes put in place to protect and save collection data.

Future efforts will focus on augmenting hardware components with outdoor equipment in order to increase the number of environments the system can operate in. This will include equipping the system will LoRa hardware to handle remote deployments and assembling a deployment kit for ease of transportation and setup. The existing system components provide a rigid foundation for expansion, and with each iterative improvement and alternate use case, the system will grow to meet the demands of the experiments it is applied to.

Appendix A

Acurite Sensor Details

A.1 Acurite 5n1 Weather Station Details

Temperature Range	-40 to 158 degrees Fahrenheit; -40 to 70 degrees Celsius
Temperature Accuracy	+/- 2 degrees Fahrenheit
Humidity Range	1% to 99% Relative Humidity
Humidity Accuracy:	+/- 5% from 0% to 10% Relative Humidity +/- 4% from 10% to 20% Relative Humidity +/- 3% from 20% to 80% Relative Humidity +/- 4% from 80% to 90% Relative Humidity +/- 5% from 90% to 100% Relative Humidity
Wind Speed	0 to 99 mph; 0 to 159 kph
Wind Speed Accuracy	+/-2 mph below 10 mph +/- 3 mph from 10 to 30 mph +/- 4 mph from 30 to 50 mph +/- 5 mph from 50 to 99 mph
Rain Gauge Accuracy	+/- 0.05-inches per inch of rainfall
Wireless Range	330 feet / 100 meters depending on home construction materials
Wireless Signal	433 MHz
Power	4 AA alkaline or lithium* batteries
Data Reporting	**Wind Speed**: 18 second updates **Wind Direction**: 36 second updates **Temperature and Humidity**: 36 second updates
Dimensions	10.8-inches Height x 5.6-inches Width x 13.8-inches Depth

A.2 Acurite Tower Sensor Details

Temperature Range	-40° to 140° F
Temperature Accuracy	+/- 2 degrees Fahrenheit
Humidity Range	1% to 99% RH
Humidity Accuracy	+/- 5% from 0% to 10% Relative Humidity +/- 4% from 10% to 20% Relative Humidity +/- 3% from 20% to 80% Relative Humidity +/- 4% from 80% to 90% Relative Humidity +/- 5% from 90% to 100% Relative Humidity
Wireless Range	330 feet; 100 meters depending on home construction materials
Wireless Signal	433 MHz
Power	2 AA alkaline or lithium* batteries (not included)
Data Reporting	16 second updates
Dimensions	4.8-inch Height x 1.6-inch Width x 0.9-inch Depth

Appendix B

Additional Sensors

B.1 Types of Supported Sensors

- Temperature and humidity sensors
- Personal weather stations
- Door and window magnetic motion sensors
- TV remotes
- Car keys
- Water sensors
- Smoke detectors
- Sound and vibration sensors
- Anything that chirps on 433MHz!

Bibliography

[1] Emmet Leffel Jonathan Goebel-Matt Perry Dragos Vas Dayne Broderson Richard Liesen Alexander Zhivov Bjorn Oberg Angela Urban. "Thermal Energy System Resilience: Thermal Decay Test (TDT) in Cold/Arctic Climates, Part I Data Collection and Protocol". In: *In press, Transactions of the American Society of Heating, Refrigerating and Air-Conditioning Engineers* ().

[2] C. Choi et al. "LoRa based renewable energy monitoring system with open IoT platform". In: *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. 2018, pp. 1–2.

[3] Brandy Diggs-McGee Dr. Alexander Zhivov ERDC/CERL Dr. Richard J. Liesen Brianna Morton. "Thermal Energy System Resilience: Thermal Decay Test (TDT) in Cold/Arctic Climates, Part II Modeling". In: *In press, Transactions of the American Society of Heating, Refrigerating and Air-Conditioning Engineers* ().

[4] Aditya Gudipati et al. "SoftRAN: Software Defined Radio Access Network". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. HotSDN '13. Hong Kong, China: Association for Computing Machinery, 2013, 25–30. ISBN: 9781450321785. DOI: 10.1145/2491185.2491207. URL: https://doi.org/10.1145/2491185.2491207.

[5] *Image: LoRaWAN Architecture*. The Things Network. URL: https://www.thethingsnetwork.org/docs/lorawan/architecture.html.

[6] O. Khutsoane, B. Isong, and A. M. Abu-Mahfouz. "IoT devices and applications based on LoRa/LoRaWAN". In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. 2017, pp. 6107–6112.

[7] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke. "A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements". In: *IEEE Access* 5 (2017), pp. 1872–1899.

[8] Benjamin Larson. *RTL_433*. Version 19. Aug. 2019. URL: https://github.com/merbanan/rtl_433.

[9] M. Rizzi et al. "Evaluation of the IoT LoRaWAN Solution for Distributed Measurement Applications". In: *IEEE Transactions on Instrumentation and Measurement* 66.12 (2017), pp. 3340–3349.

[10] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. "A survey on LPWA technology: LoRa and NB-IoT". In: *ICT Express* 3.1 (2017), pp. 14–21. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.03.004. URL: http://www.sciencedirect.com/science/article/pii/S2405959517300061.

[11] Vijendra Singh Tomar and Vimal Bhatia. "Low Cost and Power Software Defined Radio Using Raspberry Pi for Disaster Effected Regions". In: *Procedia Computer Science* 58 (2015). Second International Symposium on Computer Vision and the Internet (VisionNet'15), pp. 401–407. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.08.047. URL: http://www.sciencedirect.com/science/article/pii/S1877050915021584.

[12] J. Wan et al. "Software-Defined Industrial Internet of Things in the Context of Industry 4.0". In: *IEEE Sensors Journal* 16.20 (2016), pp. 7373–7380.

[13] A. J. Wixted et al. "Evaluation of LoRa and LoRaWAN for wireless sensor networks". In: *2016 IEEE SENSORS*. 2016, pp. 1–3.